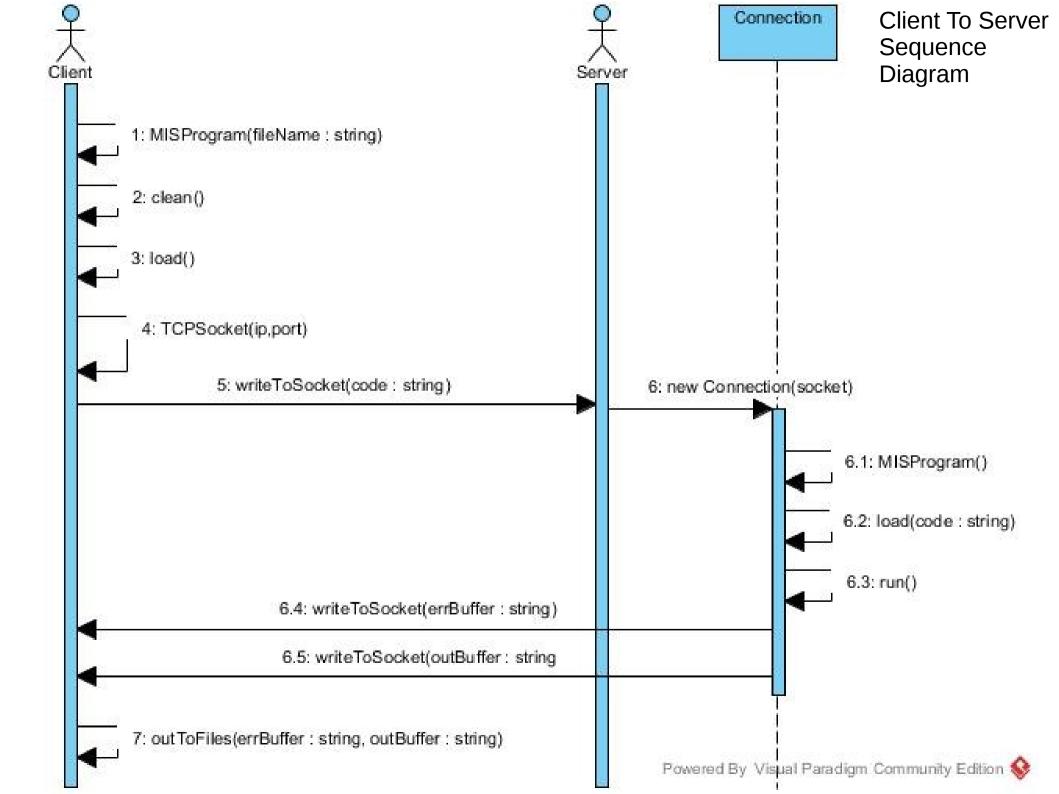


MISProgram Instruction Class Diagram -fileName : string myProgram: MISProgram' errFileName : string lineNumber : int outFileName : string parameters : vector<Parameter*> variables : map<string. Parameter*> -ptrExecute : InstructionFunction -JMPFlag : bool -instructions : vector<Instruction*> microInstructions : vector<MicroInstruction> error : bool outBuffer: string -ZFlag : bool InstructionFunction is a type of function pointer that points to a function -NFlag : bool errBuffer : string that returns void and takes in instruction : Instruction* -outBufferMutex : pthread mutex t parseParameters(paramStrings, variables, outFileName) errBufferMutex : pthread_mutex_t typeCheckParams(instructionName : string) InstructionFunctions holds all of the different functions -threadManager : ThreadManager* +e vecute() that follow the InstructionFunction format variableLocks : VariableLock* Instruction(p_lineNumber, paramStrings, p_ptrExecute, tempLoadVars) -loadLine(tempLoadVars : TempLoadVars *) +aetLineNumber(): int InstructionFunctions +getProgram(): MISProgram * -link(tempLoadVars : TempLoadVars *) +getParameters(): vector<Parameter*> * -commonLoad(inFile: T &, outToBuffer: bool) +add(instruction : Instruction *) +MISProgram() +getMicroInstructions(): vector<MicroInstruction> * +sub(instruction : Instruction *) +MISProgram(filename : string) +qetJMPFlag(): bool +div(instruction : Instruction *) +setJMPFlag(flag : bool) +clean() +mul(instruction : Instruction *) +getZFlag(): bool +assign(instruction : Instruction *) +load() +load(fileContents : string &) +setZFlag(flag : bool) +out(instruction : Instruction *) +run() getNFlag(): bool +setStrChar(instruction : Instruction *) +out() setNFlag(flag : bool) +getStrChar(instruction : Instruction *) +errorHappened(): bool -Instruction() sleep(instruction : Instruction *) +getOutBuffer(): string * imp(instruction : Instruction *) +getErrBuffer(): string * impZ(instruction : Instruction *) +qetOutBufferMutex(): pthread_mutex_t* impNZ(instruction : Instruction *) +getErrBufferMutex() : pthread_mutex_t * +impGT(instruction : Instruction *) +getVariables(): map<string, Parameter*>* +jmpLT(instruction : Instruction *) +getInstructions(): vector<Instruction*> * jmpGTE(instruction : Instruction *) +getVariableLocks(): vector<VariableLock*>* impLTE(instruction : Instruction *) **TCPServerSocket** +getThreadManager(): ThreadManager * +THREAD BEGIN(instruction : Instruction *) -sock : int -MISProgram() +THREAD_END(instruction : Instruction *) serverAddr : sockaddr in +LOCK(instruction : Instruction *) -clientAddr : sockaddr in Parameter +UNLOCK(instruction : Instruction *) -address : char type : char +BARRIER(instruction : Instruction *) -port : int Variable<T> isConstant : bool -backlog : int value : T +Parameter() +TCPServerSocket(address: char*, port: int, backlog: int) MicroInstructions holds all of the different functions +Variable() +getValue(): T & +initializeSocket(): bool setValue(value : T) hat follow the MicroInstruction format +Variable(p_type, p_isConstant, p_value) +getConnection(): TCPSocket * +getValue(): T & +getType(): char +TCPServerSocket() +getIsConstant() : bool MicroInstructions setValue(value : T) +initialize(paramStrings : vector<string> &) +initialize (paramStrings: vector<string> &) +ADDR(instruction : Instruction *) +clone(paramStrings : vector<string> &) : Parameter* +clone(param Strings: vector<string> &): Parameter* +ADDN(instruction : Instruction *) ~Parameter() ~Variable() +ADDAR(instruction: Instruction*) **TCPSocket** +ADDAN(instruction : Instruction *) remote address:char[] +ADDAF(instruction : Instruction *) my address : char[] +SUBR(instruction : Instruction *) myAddr : sockaddr in +SUBN(instruction : Instruction *) -port : int +MULR(instruction : Instruction *) sock : int +MULN(instruction : Instruction *) -peerDisconnected : bool +MULAR(instruction : Instruction *) rshutdown : bool +MULAN(instruction: Instruction*) -wshutdown : bool +DIVR(instruction : Instruction *) stream out size : long +DIVN(instruction : Instruction *) ThreadManager +TCPSocket(sock, address, port, readBufferSize, writeBufferSize) head : ThreadNode +TCPSocket(address, port, readBufferSize, writeBufferSize) tail: ThreadNode* +getRemoteAddress() : char * +getMvAddress() : char * +ThreadManager() readFromSocket(buffer : char *, maxBytes : int) +addThread(thread : Thread *) +readFromSocketWithTimeout(buffer, maxBytes, timeoutSec, timeoutMicro) cleanup() Thread ChildThread +writeToSocket(buffer : char *, maxBytes : int) terminate() -ThreadManager() pthread_attr : pthread_attr_t instruction : Instruction* +setPeerDisconnected(value : bool) +isPeerDisconnected(): bool cpu count : long +ChildThread(p_instruction : Instruction *) +shutdownRead() othread : othread t +threadMainBody(arg : void *) : void * identifier : char[] +shutdownWrite() +aetInstruction(): Instruction * Connection 1:1 +shutdown() started : bool +ChildThread() program : MISProgram* 2:1 +getStreamOutSize(): long running : bool tcpSocket : TCPSocket* +TCPSocket() mutex : pthread mutex t +Connection(p_tcpSocket : TCPSocket *) termination request : bool +threadMainBody(arg : void *) : void * ThreadNode setRunning(_running : bool) +aetSock(): TCPSocket * thread : Thread +Thread(_threadRoutine) +Connection() next : ThreadNode +isRunning(): bool +getThreadHandler(): pthread_t * +threadMainBody(arg : void *) : void * +run(arg : void *) : void * +waitForRunToFinish() getThreadIdentifier(): char * +isAlive(): bool +Thread() Powered By Visual Paradigm Community Edition &



Allocation of Program Sequence Diagram

